DOI: 10.34185/1991-7848.itmm.2023.01.073

СУЧАСНІ МЕТОДИ АНАЛІЗУ ТА УДОСКОНАЛЕННЯ ПРОГРАМ НА ОСНОВІ ГРАФОВИХ МОДЕЛЕЙ

Шинкаренко В.І., Поліщук І.А.
Український державний університет науки і технологій, Україна

Вступ. Аналіз текстів програм на основі їх графових моделей – засіб для виявлення та виправлення помилок, оптимізації та підвищення якості програмного продукту, який використовується на етапах компіляція, верифікація та тестування програмного забезпечення.

Комп'ютерні програми складаються зі складних структур, які можуть бути представлені у вигляді графових моделей. Використання графових моделей дозволяє зберігати інформацію про структуру програми та взаємозв'язки між її елементами, що допомагає підвищувати якість програмного продукту та зменшувати ризики при зміні функціоналу.

Основний матеріал. Зазвичай графові моделі програм складаються з двох частин: Граф потоку управління (CFG) і Граф потоку даних (DFG).

Граф потоку даних (DFG) – це модель програми без умовних операторів. DFG включає в себе вершини, які представляють обчислювальні операції, та ребра, які вказують на потік даних між цими операціями. Наприклад, якщо результат операції A використовується у операції B, то між цими вершинами в графі буде ребро.

DFG відображає порядок, у якому виконуються операції в коді. Це одна з переваг графа потоку даних. Мається можливість його використання для визначення можливих змін порядку операцій, що може допомогти зменшити конфлікти конвеєрів або кешу. DFG можна використовувати, коли точний порядок операцій просто не має значення. Граф потоку даних визначає часткове впорядкування операцій у базовому блоці. [1]

Приклад графу потоку даних наведено на рисунку 1.

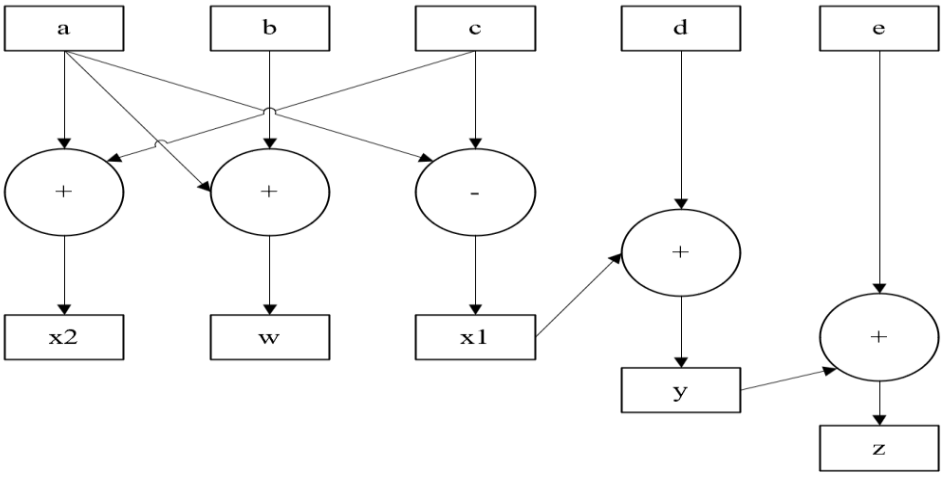

Рисунок 1 – Граф потоку даних

Граф потоку керування (CFG) – в теорії компіляції – множина всіх можливих шляхів виконання програми, поданих у вигляді графа. [2]

У CFG вузлами є базові блоки програмного коду, а ребрами – взаємозв'язки між ними, якими є умовні або безумовні переходи виконання.

За допомогою графа керування можна визначати недосяжні фрагменти коду, деякі види зациклення програм, можливість перегрупування операторів та інше. [2]

Приклад графу потоку керування наведено на рисунку 2.

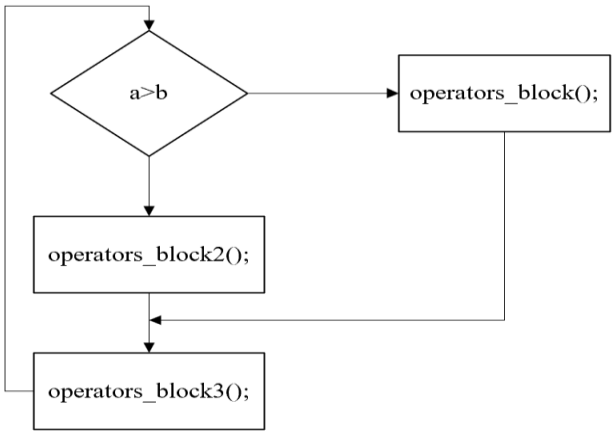

Рисунок 2 – Граф потоку керування

Граф потоку керування та даних (CDFG) – це комбінація CFG та DFG, яка відображає і порядок виконання операцій у програмі, і потік даних між ними. У CDFG ми маємо два типи вузлів: вузли прийняття рішень і вузли потоку даних. Вузол потоку даних інкапсулює повний граф потоку даних для представлення

базового блоку. Ми можемо використовувати один тип вузла рішення для опису всіх типів керування в послідовній програмі. [1]

CDFG може бути використаний для виявлення різноманітних проблем, таких як помилки в потоці даних, нескінченні цикли, недосяжні блоки коду та інше.

У роботі [3] розглянуто проблему автоматичного ремонту програм на основі діагностичних відгуків (наприклад, повідомлень про помилки компілятора).

Спочатку створюється граф програмного зворотного зв'язку, який з'єднує символи, пов'язані з відновленням програми у вихідному коді та діагностичним зворотним зв'язком, а потім застосовуємо нейронну мережу для моделювання процесу мислення. По-друге, застосовується парадигма контрольованого навчання для відновлення програм, яка використовує непомічені програми, доступні в Інтернеті, для створення великої кількості додаткових прикладів відновлення програм, які використовуються для попереднього навчання моделей [3] .

Автоматичне відновлення програми має потенціал для значного підвищення продуктивності програмування. Зокрема, поширеним джерелом програмних помилок є помилки компілятора, які включають використання невирішених символів, відсутні роздільники (наприклад, дужки) і помилки типу. Ці помилки зазвичай спостерігаються як у програмістів-початківців, так і у професійних розробників, а також у прогнозованому коді синтезу програм. Відповідно, використання машинного навчання для виправлення помилок компілятора останнім часом викликало значний інтерес.

Застосування графових моделей CDFG для виявлення плагіату має кілька переваг. По-перше, вони дозволяють врахувати не тільки синтаксичні відмінності між програмними кодами, а й їх семантику, тобто спосіб, яким код виконується. По-друге, вони можуть допомогти зменшити кількість помилкових виявлень плагіату, оскільки графові моделі дозволяють враховувати контекст коду. Наприклад, інструкції, які виконуються в різних частинах програми, можуть мати різні семантичні значення, і тому не слід розглядати їх як копії.

Одним з можливих підходів для зменшення кількості неточних виявлень плагіату є перестроювання графів. При цьому структура графа може бути змінена з метою підвищення точності порівняння. Крім того, можливо використовувати методи машинного навчання, щоб покращити точність графових моделей CDFG. Наприклад, можна навчити модель виявляти певні

властивості графових моделей, які допоможуть покращити їх точність та ефективність. Такі підходи можуть допомогти зменшити кількість помилкових виявлень плагіату і покращити його ефективність.

Порівняння графів різних програм є корисним інструментом для виявлення збігів та відмінностей у логіці роботи програм, яке може допомогти виявити неефективність та помилки в програмі, а також запропонувати нові або покращені функції на основі знаходжень у графах інших програм.

Одна з можливостей порівняння графів полягає у виявленні схожостей між декількома програмами. Якщо дві програми мають подібну логіку роботи, то їх CDFG може мати подібний вигляд. Це може бути використано для виявлення збігів у функціональності програм та для виявлення спільних елементів, які можуть бути використані для створення нової програми. Наприклад, якщо програма А має швидкодію нижче, ніж програма Б, то CDFG програми А може містити деякі елементи, яких немає в CDFG програми Б. Виявлення цих різниць може допомогти програмістам виявити неефективність та помилки в програмі та запропонувати нові або покращені функції на основі знаходжень у графах інших програм.

Висновки. Питання аналізу тексті програм потребують додаткового дослідження, проте застосування графових моделей CDFG для виявлення плагіату та удосконаленню програм може бути перспективним напрямком дослідження. Це може допомогти покращити етику програмування та забезпечити доброчесність у процесі розробки програмного забезпечення.

Література

1. Marilyn Wolf. Computers as Components // Morgan Kaufmann, 2016. – 545 с
2. Граф потоку керування: [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Граф_потоку_керування
3. Michihiro Yasunaga. Graph-based, Self-Supervised Program // Michihiro Yasunaga, Proceedings of Machine Learning Research // 2020. – 10 с
4. Zhongda Yuan. Automatic enhanced CDFG generation based on runtime instrumentation. / Zhongda Yuan, Yuchun Ma. – IEEE 2013. – 15 с.

MODERN METHODS OF ANALYSIS AND IMPROVEMENT OF PROGRAMS BASED ON GRAPH MODELS

Shynkarenko Viktor, Polishchuk Illia

Abstract. The approach of automated computer program code plagiarism detection and enhancement based on reconstruct able CDFG graph models was proposed. It is suggested to use CDFG graph mode for analyze the computer

program texts. The program plagiarism detection and enhancement can be done based on the similar program graph model analyses and different graph operation from the graph theory. The usage of CDFG-graph model allows to take in account both program data and control paths. This can help improve programming ethics and ensure integrity in the software development process.

Keywords: graph models, computer programs, CFG-graphs, DFG-graphs, SDFG-graphs, program plagiarism.

Reference

1. Marilyn Wolf. Computers as Components // Morgan Kaufmann, 2016. – 545 c.
2. Control-flow graph: [Wev resource]. – Access: https://en.wikipedia.org/wiki/Control-flow_graph
3. Michihiro Yasunaga. Graph-based, Self-Supervised Program Repair // Michihiro Yasunaga, Proceedings of Machine Learning Research // 2020. – 10 c.
4. Zhongda Yuan. Automatic enhanced CDFG generation based on runtime instrumentation. / Zhongda Yuan, Yuchun Ma. – IEEE 2013. – 15 c.